5. MAN-RATED FLIGHT SOFTWARE FOR THE F-8 DFBW PROGRAM

Robert R. Bairnsfather
The Charles Stark Draper Laboratory, Inc.

SUMMARY

The design, implementation, and verification of the flight control software used in the F-8 DFBW program are discussed. Since the DFBW utilizes an Apollo computer and hardware, the procedures, controls, and basic management techniques employed are based on those developed for the Apollo software system. Program Assembly Control, simulator configuration control, erasable-memory load generation, change procedures and anomaly reporting are discussed. The primary verification tools—the all-digital simulator, the hybrid simulator, and the Iron Bird simulator—are described, as well as the program test plans and their implementation on the various simulators. Failure-effects analysis and the creation of special failure-generating software for testing purposes are described. The quality of the end product is evidenced by the F-8 DFBW flight test program in which 42 flights, totaling 58 hours of flight time, were successfully made without any DFCS inflight software, or hardware, failures or surprises.

INTRODUCTION

From early 1971, CSDL participated in Phase 1 of the Digital Fly-by-Wire program being administered by NASA Flight Research Center (NASA/FRC). Overall program effort was directed toward a series of demonstration Fly-by-Wire (FBW) aircraft flights. A triply redundant Analog Fly-by-Wire (AFBW) Backup Control System (BCS), employing a simple open-loop control algorithm, is coupled with the primary flight control system to provide the two-fail-operate/fail-safe reliability necessary for severing mechanical linkages. The simplex Digital Fly-by-Wire (DFBW) Primary Control System (PCS) has both software and hardware failure-detection capability in the digital computer. There are also independent monitoring and failure-detection modules operating on PCS control commands, power supplies, pilot input devices, and other critical areas. Finally, there is the capability for pilot-initiated downmoding to BCS via several independent paths. There are seven selectable PCS flight control modes available. Three Direct (DIR) modes consist of pilot stick/pedal plus trim applied directly to the control surfaces. Three Stability Augmented System (SAS) modes incorporate body-axis angular rates (and lateral acceleration) as feedback variables. The Command Augmented System (CAS) mode is basically pitch SAS with normal acceleration feedback and forward-loop integral bypass. The only BCS mode, Direct, is also selectable by axis.

The first Fly-by-Wire flight was made on 25 May 1972, in the high performance F-8C fighter assigned to the DFBW program. Takeoff and landing were made in PCS/DIR. Basic performance and handling qualities were demonstrated at several flight conditions, both in BCS and PCS/DIR. Closed-loop PCS/SAS was first flown on 18 August 1972 with subsequent flights building toward full system capability. The demonstration flight test program continued through late 1973.

The CSDL role in the F-8 DFBW program has been directed at the PCS software, hardware, and peripherals. Specific tasks have been: the hardware design, development, and testing of the uplink and downlink converters, the PIPA Simulator, and the Gimbal Angle Simulator; and software design, implementation, and verification of the NASA/FRC three-axis Primary Control System algorithms; the functional design, software design, production, and verification of the mode and gain change routines, miscellaneous ground test programs, and open-loop inflight earth-rate torquing routine; the interface design including failure analysis; simulation support; the review and verification of preflight erasable loads.

The F-8 DFBW System

Aircraft—The F-8C Crusader, a carrier-based U.S.Navy fighter of mid-50's vintage, is a high-performance single-seat aircraft capable of Mach 1.8 flight at altitudes of 60,000 feet. NASA/FRC obtained several surplus aircraft of the F-8 series. Two of them are involved in the F-8 DFBW program, one as the flight article and one as the Iron Bird Simulator test article. Figure 1 depicts the F-8C aircraft, showing the physical distribution of key F-8 DFBW hardware. Descriptions of the hardware are given in Table 1 and Table 2.

Digital System—The digital computer used by the PCS is the general purpose Apollo/LM Guidance Computer (LGC). An Apollo Inertial Measurement Unit (IMU) provides attitude angles, angular rates, and linear accelerations for feedback control. Major considerations for using the Apollo hardware were that it possessed a demonstrated reliability and flexibility. Moreover, surplus LM hardware was available from cancelled Apollo missions. Experienced teams of software and hardware specialists were also available, for software and systems integration tasks, at CSDL and Delco Electronics. A functioning Operating System software existed for the LGC, in addition to the supporting facilities of the powerful Assembler software, the All-Digital Simulator, and two hardware-integrated simulators at CSDL. Starting with this framework meant that a significant portion of the development task was already completed. There were some disadvantages, the most significant being the July 1972 scheduled shutdown of the core-rope manufacturing facilities for the LGC fixed memory. Another disadvantage, although not recognized immediately, was that the F-8C performance envelope exceeded the design capabilities of some Apollo hardware items. This influenced the digital flight control system (DFCS) performance, and required a reduced performance envelope, which, while less than F-8C capabilities, was nevertheless acceptable for an experimental digital fly-by-wire testbed.

Computer—The LGC contains two distinct memories, fixed and erasable, as well as hardware logic circuits. The fixed memory is stored in a wire braid which is manufactured and installed in the computer. This memory cannot be changed after manufacture and it can only be read by the computer. Fixed memory contains 36,864 words of memory grouped into 36 banks. Each word contains 15 bits of information, plus a parity bit. The erasable memory makes use of ferrite cores which can be both read and changed. It consists of 2048 words divided into 8 banks. Erasable memory is used to store such data as may change up to or during a mission, and is also used for temporary storage by the programs operating in the computer. The memory cycle time (MCT) in the LGC is 11.7 μs. Most single-precision instructions are completed in two MCTs; most double-precision machine instructions are completed in three MCTs.

SOFTWARE DEVELOPMENT

The software control procedures employed for F-8 DFBW selectively follow those developed and successfully applied during the generation of software program assemblies for the Apollo command and lunar module computers. A continuation of useful procedures, made necessary because the F-8C uses the same Apollo hardware, and desirable because of schedule limitations, was easily imposed by the CSDL personnel connected with F-8, all of whom were contributors to the Apollo effort. The limited scope of F-8 dictated some changes in procedure, but these were basically simplifications commensurate with the level of effort. After all, approximately 400 man-months/month were expended in Apollo by CSDL programming and engineering groups just prior to the first lunar landing, while F-8 DFBW peaked at about 9 man-months/month. The critical time span was from Control Law Specification delivery in March of 1971 until program release for fixed-memory core-rope manufacture in mid-December of 1971. Since that date, CSDL has supported Preflight Erasable Load generation, failure analysis, preflight procedure preparation, and Erasable Memory Program development and verification. The timely development and excellent flight-test performance of DFBW software attest to the effectiveness of the control procedures employed. It is worth emphasizing that we now have more modern software techniques, but that Phase 1 of F-8 DFBW was a basic evaluation program, and utilized off-the-shelf software as well as hardware. Approximately 85 man-months and 95 hours of IBM 360/74 computer time were required for the Phase 1 software design, implementation, and verification tasks. The F-8 chronology is shown in Fig. 2.

Operational Software

The operational software for F-8 DFBW consists of two basic categories: the DFCS Program Assembly, and the Preflight Erasable Load Assembly. In the finished product, the DFCS Program Assembly is embodied in the core rope and comprises the computer's fixed memory. At this stage, it has become hardware and is effectively a breadboard autopilot in that the structure is invariant while most parameter values and switch words are variable. For F-8 DFBW, there is only one final Program Assembly, from which the flight rope and an identical

spare are manufactured. The Preflight Erasable Load Assembly is embodied in a tape and comprises the computer's Initial Data Load. The tape, KSTART, contains parameter values and switch settings required by the program, and the computer receives it as a part of each power-up sequence. A new Preflight Erasable Load Assembly is made whenever a flight test requires new parameter values. To ensure the high degree of reliability and safety that is necessary for man-rated flight software, both assembly processes are carefully controlled.

Program Assembly

The Program Assembly has two main functional areas: Systems and Applications. Grouped under Systems are Executive, Restart, and Service. Applications covers Flight Control, and Miscellaneous. The Executive code includes the priority job-queue processor, the time task-queue processor, the time-dependent interrupt processor, the idle-job routine. The Restart code includes the hardware restart interrupt processor, computer initialization routine, the program alarm processor, the restart-group phase-control routines. The Service code includes the list-processing interpreter, the IMU monitor, the computer self-test routines, the man-machine interface routines, the interrupt processors. The Flight Control code includes the autopilot initialization routine, the mainline processor, the filter pushdown and wrap-up processor, the input discrete processor, the Mode and Gain change processor, the body transformation matrix processor. The miscellaneous code includes the ground test programs, and special-purpose applications routines.

In several areas, the flight control requirements and the LGC characteristics posed interesting problems. Some of these are singled out.

Duty Cyle—Early in the development process it became clear that the Flight Control system would create a relatively high duty cycle in the LGC due to several causes: LGC instruction time (24 μs/instr), the flight control sample period (30 ms) and the generalized nature of the control system. Since the entire LGC is devoted to the DFCS, words of code could be traded for increased time efficiency wherever possible; that is, code is designed for minimum execution time rather than for minimum storage. Time savings are also realized for control parameters, where combinable multiple parameters are replaced by an equivalent single parameter in a working register, whose value is generated only once by program initialization.

Restart Protection—A hardware restart is a special interrupt that takes precedence over all other interrupts, and that cannot be inhibited. The hardware restart is triggered by circuitry in event of selected computer malfunctions. On completion of the restart, all output channel discretes are cleared, and computer control is transferred to a specific memory location, i.e., to the Restart Routine. The Restart software rapidly reestablishes the channel output interfaces because F-8C control surface commands and the PCS primary-enable signals depend on a viable interface. The restart software next restores the program flow by reestablishing the job-queue and time-queue, and by causing the program whose execution was interrupted to resume at the latest restart

point. Restart points are entry points, breaking program flow into separate blocks, such that a properly restart-protected program will reproduce the same values after a restart as before.

In general, a repetition of code execution is involved following a restart because the nature of the LGC requires software recovery procedures. However, the repetition requires that special care be taken during code generation to avoid creating situations where a restart will cause a multiple update of a variable. For example, if the operation A+B → A occurs between two restart points, then A is updated at each pass through the code. This violates the rule that the values generated by code repetition after a restart must be the same as before. The situation of multiple updates is avoided by a copy cycle, which involves an intermediate variable and an additional restart point. For the example we have A+B → C, followed by the new restart point, followed by C → A. Clearly, the final value of cell A is unaffected by code repetition. Copy cycles are common in Apollo code and have the advantage of economy of erasable memory usage although they are expensive in terms of execution time. Note that cell C is intermediate and can be used by many copy cycles.

Rather than use copy cycles, F-8 DFBW prefers a method that, because of the high DFCS duty cycle, is conservative of time but is expensive in fixed and erasable memory cells, doubling the number. Two functionally identical strings of code, a J-branch and a K-branch, are required with processing alternating from one to the other. Two equivalent sets of erasables are required, also J-branch and K-branch. The J-branch code uses K-branch (past value) outputs plus J-branch (present value) inputs to compute J-branch (present value) outputs. No special copy cycles are required, and computations are efficiently performed. Copy cycles would likely have pushed DFCS duty cycle dangerously close to 100%. It reaches 91% even with time-efficient restart protection.

Indirect Transfer—At sixteen critical points in F-8 DFBW program flow, and at one point in the downlink program, a capability is provided for erasable indirect transfer of control. In application the program flow of the hardware core-rope fixed memory program is determined by the address contained in a specific erasable cell at the time the cell is accessed by the program. Erasable cells used in this manner fall into two classes. There is the class of cells whose contents (the destination address) is changed regularly under program control, say every 20 ms or 30 ms. These cells, although erasable, form an integral part of the core-rope. The second class consists of cells whose contents are in general established only once, either by an initialization pass or by the Initial Data Load (KSTART tape). It is this second class of erasable cells that provides the powerful capability of altering the program flow after core-rope manufacture by means of Erasable Memory Programs.

Generalized Filters—Inasmuch as F-8 DFBW is a flying breadboard, the feedback sensor quantities are each provided with a generalized filter. The five filters, three for body rates and two for linear accelerations, allow flexibility of filter choice: bypass, first order, second order, and third order. An alternate third order is obtained by cascading the first and second

order sections to obtain control over individual poles and zeros. The filter coefficients are parameters in the KSTART tape. The filters are active at all times, even in BCS/DIR.

The computations are divided into two phases, the main phase which incorporates the current input with past values to update the output, and the pushdown or wrap-up phase which updates the other filter quantities in preparation for the next cycle. In this manner the control surface commands which use the filter outputs are generated with the shortest delay. The time-consuming filter wrap-up calculations are not performed until after closing the aircraft control loop, and so do not contribute to the delay. The saving is significant because the wrap-up can represent as much as 92% of the total filter load.

Gain Change—Manual gain changing is provided in lieu of automatic gain changing as a function of, say, dynamic pressure. Separate pitch, roll, and yaw gain-select switches on the MAPP, each with four positions, comprise the pilot interface. Selection of a specific gain (or coefficient) parameter is made from a fixed list of 105 candidates, serially numbered from 1 to 105. Each gain is associated (by axis) with a particular gain-select switch, and a maximum of 9 gains can be designated for a given flight. Each gain chosen, with its serial number and four values, becomes part of the PEL. When a gain-select switch is changed by the pilot, the program recognizes the change and the PEL-designated gains associated with that switch axis are changed. For each gain in turn, a small routine implements the change, performing all necessary scaling, recomputing all working registers using that gain, and initializing any filter using that gain.

Erasable Memory Programming—Erasable memory programming provides the only means of modifying the program once the core rope is manufactured. Modification can sometimes be accomplished by breaking into the program flow at a suitable erasable branch point, which must be of the second class as defined above. The procedure is to change the erasable cell contents to point to an unused block of erasable memory and to load executable code into that area (called an Erasable Memory Program or EMP). The final instruction of the EMP returns control to the fixed memory program. The EMP allows some unanticipated problems to be solved by shoehorning suitable code into the program flow.

Erasable Downlist—In Apollo, the identification and specification of telemetered data was done by means of address tables built into the core rope. For a mature design such as Apollo, quantities of interest are well known, and properly can be built into the rope. F-8 DFBW, on the other hand, must offer flexibility for experimental design. Variables and quantities of interest can change from day to day depending on a given flight plan. To accomplish this end, erasable specification of the downlist quantities by means of KSTART tape is incorporated into the Downlink program.

Preflight Erasable Load Assembly

Flexibility is achieved in the F-8 DFBW despite the hardware status of the core-rope program by providing for a large number of erasable parameters. The aggregate, called the Preflight Erasable Load, consists of three categories: Data words, Downlist Words, and Erasable Memory Program words. The Data words are constants and include loop gains, filter coefficients, nonlinearity parameters, IMU compensation parameters, branch control parameters, and branch control address constants. The Downlist words are address constants that define the quantities to be telemetered. The EMP words are executable code and associated constants.

Early in the program the Preflight Erasable Load and the KSTART tape consisted only of Data words and Downlist words, and were generated by CSDL. But the responsibility for the data values resided with FRC, so generation of the Preflight Erasable Load and KSTART shifted to FRC as the software capability was developed there. However, Erasable Memory Program development was a CSDL function, and the verified and accepted EMP code was incorporated into the KSTART by FRC.

Several unique or extremely helpful features characterize the F-8 Preflight Erasable Load (PEL), and the generation of its KSTART uplink tape, specifically:

(1) PEL parameters are expressed in conveniently scaled, physically significant engineering units.

(2) A DFCS initialization routine translates each PEL parameter (units and scaling) into DFCS operational parameters. Factored or ratioed parameters are combined into single operational parameters at this time.

(3) Comprehensive error checking and diagnostic indicators are built into the KSTART tape generating programs.

Parameters—The basic DFCS parameters are expressed in conveniently scaled engineering units and constitute the erasable load. The DFCS working registers (gains, limit levels, coefficients) are defined so as to minimize computation time where possible. This usually results in unusual scaling, e.g., number of DFCS samples instead of seconds, or DAC bits instead of surface degrees. Other working registers are functions of basic parameters, such as a simple product, or a limit level that is computed from intercept/slope/breakpoint values. Also a working register might contain an address constant, selected from a table in accordance with certain rules. To accomplish the interface between working registers and erasable load parameters, F-8 DFBW utilizes an initialization routine. By having an initialization routine available to translate the working registers, the engineer preparing KSTART tapes, or changing parameters manually via the DSKY during a simulation, can continue to think in basic engineering terms. This is especially important in F-8 DFBW, since much of the development is performed on hybrid simulators

where the DSKY interface is the only practical interface for changing DFCS parameters. By keeping PEL specifications simple and by formulating them in engineering terms for both physical feel and visibility, the possibility for error is greatly reduced. Since programmed and verified initialization software is involved, reliable and complete changes are made quickly by single-parameter data entries even though that parameter exhibits multiple usage.

KSTART Generation—Two off-line diagnostic programs, DOWNDIAG and SHERLOCK, developed by NASA/FRC, contribute significantly to the generation of a highly reliable PEL and its KSTART tape. Operational use of these programs is shown schematically in Fig. 3.

DOWNDIAG checks the erasable downlink list specification against format, opcode, address, and keypunch errors. It punches the Erasable Downlist (EDL) and Downlink Processor (DLP) decks only after error-free input is provided. The DLP deck is used for post-flight or post-simulation downlink processing. The EDL deck is integrated with the DFCS parameter deck for input to SHERLOCK.

SHERLOCK likewise checks against keystroke, octal, and address errors, but more significantly performs comprehensive reasonability checks, e.g., minimum/maximum range or compatibility between related elements. SHERLOCK also extracts filter polynomial roots, checks the stability of poles, and checks zeroes against minimum/maximum ranges. Diagnostic printouts must be answered by corrections to the SHERLOCK inputs, or by signed waivers, before output decks are punched, one for the F-8 All-Digital Simulator at CSDL, and the other for input to KPUNCH, the KSTART tape diagnostic and punch program.

KPUNCH calculates the initialization values for the uplink summation (UPSUM) registers such that with a proper uplinking of the KSTART tape, the UPSUM registers equal 77777 77777 when displayed on the DSKY. Errors generated during uplinking will leave numbers other than 7s. KPUNCH also performs limited diagnostic checking and ultimately punches the KSTART tape, ready for uplinking to the LGC prior to flight.

F-8 DFBW Software Package

The F-8 DFBW software package can be broken down as in Table 3 (Fixed Memory Allocation), and Table 4 (Erasable Memory Allocation). The DFCS code is by far the largest single item. Extensive fixed memory is used by Display Interfaces (DSKY processing), Interpreter/Executive, and IMU Alignment. Most of this code was transferred directly or with minor change from the LM program for Apollo 14. The Self-Test Self-Check code came from Apollo preflight erasable code. Roughly half (696) of the erasables used are DFCS related, and a significant number (389) belong to the Preflight Erasable Load.

SOFTWARE PROGRAM CONTROL

The flight software for F-8 DFBW program leans heavily on the experience developed for Apollo. The main difference between Apollo software and other

(previous) software is that the Apollo software had to work perfectly the first time it was used in its real environment. Apollo manned missions had a one-shot nature that required guaranteed performance. To achieve such reliability, management and supervision controls were set up, and have evolved over several years into a system to monitor and check software progress very closely and yet not to create an environment that is oppressive to the creativity, perseverance, and dedication of engineers. The system thus created has been proven in both developmental and incremental phases of software. Man-rated flight software depends on reliability and confidence built up by careful management and supervision controls supported by thorough software verification using real hardware and high-fidelity models in simulation.

Software Management

A successfully managed software effort must provide:

(1) Realistic estimates of requirements including manpower, assembly and simulation budgets, memory allocations.

(2) Efficiency in the development and verification process including non-overlapping testing, effective use of man and machine resources.

(3) Achievement of milestones on schedule.

(4) Visibility of the product including developmental status, trouble spots, user-oriented operations and interfaces.

(5) Flexible and efficient response to design change requests.

(6) Systematic verification procedures at all module interface levels of testing and performance.

(7) Reliability of final products.

(8) Quality performance of final products.

The software management and control system developed for Apollo provided such capability. Its selection for F-8 DFBW was a natural outgrowth of successful prior experience with it. Changes were made, but only when the differing situations indicated a modified approach.

The management and control of flight software is directed toward the timely preparation of two end items: a software program assembly from which the read-only core-rope memory is manufactured, and a software preflight erasable-load assembly from which a KSTART tape is manufactured to initialize the erasable read-write memory. Operational efficiency, performance capability, operational flexibility, and overall reliability are demanded of both the fixed and the erasable-memory assemblies, since they complement each other in terms

of overall performance. Timely availability is likewise a requirement in terms of schedule milestones. Changes and additions to the baseline design must be implemented with the same quality and timely control.

Organization and Controls

The software organization used by F-8 DFBW is relatively simple. The Project Manager is the customer's contact point. The Project Manager interfaces with the Software Manager, who interfaces with the engineers doing the software design, coding, and verification. Both of the latter interface with Assembly Control, which is responsible for the assembly process. The types of control machinery available to the Project Manager and the Software Manager are as follows:

(1) Software Specification Document is the product specification to which the software must conform.

(2) PCR—a Program Change Request, that officially changes the Software Specification (must be signed off by customer, Project Manager, and Software Manager).

(3) PCN—a Program Change Notice, similar to a PCR but deemed imperative by CSDL (must be signed off by Project Manager and Software Manager).

(4) Anomaly—a request to fix an error in the program (must be signed off by Project Manager and Software Manager).

(5) ACB—an Assembly Control Board request, identifies a necessary program change that is not a specification change (must be signed off by Software Manager).

Under Configuration Control, all coding changes and additions must be covered by one of the above forms of approval before the Assembly Control Supervisor will incorporate the code into the assembly.

Assembly Control

The Assembly Control functions in Apollo were highly structured and very formal for the mainline program assemblies. There was an Applications Programming Development and Testing Group for the two major assemblies. A System Integration Programming Group served for all assemblies, but the major assemblies had separate Assembly Control Supervisors. Finally, the Assembly Control Service Group served all needs.

The software generation process is illustratively simplified in Fig. 4. A coding task is routed to the appropriate programming group for code design. Discussions with the other groups might follow. Completed code is submitted to Assembly Control where it is either accepted for the next revision or returned

for corrections. At appropriate times, the assembly update deck is submitted to make the new revision. The Assembler output is examined by Assembly Control and errors are either fixed or referred back to the coder for rectification. Notification of a good assembly is given to coder/testers who submit simulation test runs. If tests do not work correctly, corrected code is submitted for the next revision. On receipt of good results, a new coding task is begun.

In F-8 DFBW, with a total programming team of about nine people, such structuring was not practical or necessary. Nevertheless the spirit of the Assembly Control process was maintained. One member of the DFBW team was designated Assembly Control Supervisor, but his activities spanned all four of the structured areas as time permitted and activity made necessary. For example, he monitored, coordinated and submitted all assembly changes, maintained the Simulator test packages, published the assembly documentation, maintained and verified IGC System software, coded and verified some Applications code, and participated in Level 4/Level 5 testing. The other team members likewise found their activities spanning the four groups as specific needs came and went, each contributing in areas of greatest interest and ability.

Controllable Items

In addition to the main program assembly, there are also other areas where control procedures must apply. These are the Preflight Erasable Load Assembly, Simulator Test Packages, Off-line Program Assemblies, and Erasable Memory Programs.

A Preflight Erasable Load Assembly is associated with each mainline program revision, and consists of data constants, branch-control constants, and address constants that are defined in the mainline revision. The Preflight Erasable Load Assembly is used to generate data and address decks for Simulator test runs and it is essential that these decks be error free.

The Simulator Test Package supports the software testing and verification by providing a common library of test case decks. Functionally the decks cover three categories: program initialization, simulation control, and edit control. Operationally the decks are invoked in suitable configurations at run time by single cards in the user's test deck.

Off-line Assemblies—As the mainline program matures, off-line versions are useful to check out code prior to updating the mainline assembly. Once the design and coding is checked out, a simple transfer of appropriate code is made to the mainline assembly. In F-8 DFBW two examples occurred; one was to check out a major design modification in the BCS downmode logic just prior to Configuration Control, and the other was to create a testing and training tool capable of failing input/output discretes via DSKY commands.

Erasable Memory Programs—Erasable-memory programming is a tool enabling a limited flexibility for modifying core-rope program flow. A block of code is designed to reside in and operate from erasable memory, and a way is devised to access the code from the existing rope.

Assembly Control Tools

Assembler—Since the software was not written in a Higher Order Language, a sophisticated assembler was of utmost importance. The Assembler is by far the most powerful tool in the Assembly Control process. The lengthy evolutionary period of Apollo has generated many fine features.

Diagnostic Package—The Assembler diagnoses faulty coding in both basic and interpretive languages. It issues diagnostic messages about references to non-existent variables, multiple definitions, illegal sequences of instructions, improper erasable-bank or fixed-bank references, and many others.

Basic and Interpretive Language—The Assembler recognizes two languages: basic language, and a list-processing interpretive language. The latter permits vector and matrix as well as double and triple precision operations; these are processed by the Interpreter software routines in the LGC. The Assembler recognizes data constants, noun and verb constants, downlink list specification constants, and address constants.

Flexibility of Memory Allocation—Blocks of fixed-memory programming can be referenced to each other so that if a block expands, another block need not be moved to make room for it. Overlapping of program memory is flagged if it occurs. Overlapping of erasable storage (time-sharing), on the other hand, is facilitated by the Assembler.

Program Visibility—The Assembler provides complete mnemonic cross-reference tables, a summary of erasable memory assignments, and maps of both erasable- and fixed-memory storage. All operand references are threaded, allowing rapid eyeball debugging even when the relevant passages are scattered through hundreds of pages. Word count, including a breakdown by functional area, is provided.

Modularity—The Assembler provides the ability to separately assemble and partially diagnose sections of the full program. These can be coded separately and brought together into full programs for verification.

Interface with All-Digital Simulator—The Assembler output includes input information for the All-Digital Simulator, which is useful for simulator initializations, and for simulator run-time diagnostic error detection. The Symbol Table enables the addressing of erasable cells and fixed locations by name, rather than by number which tends to vary from revision to revision as memory layout is modified. Tapes for fixed-memory loading of core-rope simulator can be generated. Constants, bad words (assembler-detected errors), unused words, and coding instructions are distinctively flagged to permit detection of such run-time errors as 'executing a constant' or 'executing from unused fixed memory'. KSTART tapes can be punched directly from the Preflight Erasable Load Assembly as a feature of the Assembler.

Erasable Memory Map

The limited erasable-memory size of the LGC forced a policy of cell sharing as a means of extending memory capability in Apollo; extensive cell sharing was necessary, more than doubling the erasable complement and resulting in as many as seven distinct usages. An erasable-memory map was used as a bookkeeping and planning tool. The map was looked on as a short-lived necessity, otherwise the cell-sharing process would have been automated. In F-8 DFBW, even though memory cell sharing is limited, the Erasable Memory Map is an especially useful document. A separate map is prepared for each erasable bank by the Assembly Control Supervisor. The primary allocation is identified in the first column, with the overlays defined in the subsequent columns. The map simplifies the problem of assigning multiple use to cells or blocks of cells and minimizes the problem of run-time conflicts between LGC programs. The maps are extremely valuable to the programmer preparing erasable memory code by identifying unused blocks of cells and by aiding in the time-sharing usage of cells.

Software Development Activity

The software development process, involving all phases of software activity, can be summarized in Fig. 5. All software design is based on written specification. In Apollo, the specification was the seven volume Guidance System Operations Plan. In F-8 DFBW, the Control Laws, backup interface requirements, pilot interface requirements, and data retrieval requirements are prescribed in the Software Specification. The LGC executive hierarchy, service routines, interrupt processors, restart routines, downlink, and all others that came from Apollo are specified by inference as being the same as Apollo. The few changes in this category by rights should be documented by PCRs or ACBs. However the ultimate documentation in this area, as was similarly true in Apollo, is the detailed flowchart. Nevertheless, in the software development, authorization must exist in one of the forms: Software Specification, Program Change Request, Program Change Notice, or Assembly Control Board direction.

Another class of input to the Software Development, shown in Fig. 5, is the Initial Data Load which becomes the Preflight Erasable Load. The load is the cumulative array of values for control law parameters and for other routines' parameters and, as such, is jointly specified by FRC and CSDL. The load is revised and updated to keep pace with the software development.

A third class of input to the software development is the test plans, the most important one being the Level 4 Test Plan. Test plans exist at all levels and are the basis for the level testing. At the lower levels, the plans are informal tools to ensure thorough unit testing by individual programmers. The Level 3 Test Plan and the Level 4/5 Test Plan are carefully documented compendiums of specific tests, and cover all areas of the software. The test plan is reviewed and updated by all concerned; it can be added to at any time to include any overlooked areas.

Continuing in Fig. 5, the software is designed in blocks or units with each being tested before proceeding to the next. Testing at the unit level (Level 1/2) is generally bit-by-bit digital simulation. When a sufficient number of units are completed, the hardware and alarm interfaces are tested as appropriate. These tests generally involve all three simulators: the Digital, Hybrid, and System Test Laboratory. Modular Testing (Level 3) commences in any given area when all units in a given program function are completed, for example, the DFCS Direct Mode in the pitch axis. This level of testing continues until all DFCS modes and capabilities are completed. Since several program areas are developed in parallel, but not all at the same rate, testing at several levels takes place during any given time frame.

When all major programs appear to be essentially completed, Configuration Control is instituted, officially designating the start of Level 4, although limited Interface testing can take place earlier. Subsequent to Configuration Control, all program changes require the careful scrutiny and approval of one or more of the software supervisors, as well as the coding experts in the areas affected. Software Specification changes require a PCR. Level 4 tests are based on the Test Plan, and all incorrect, or unexpected, or incomplete, or anomalous behavior is documented in an anomaly report or a discrepancy report. Discrepancies are software errors detected after Configuration Control, but prior to release-for-manufacture. Anomalies are software errors detected after release-for-manufacture. Verification at Level 4 and above involves exercising the program on the three CSDL simulators, as well as the FRC Iron Bird System. All documented anomalies and discrepancies must be resolved. In some cases resolution of a Hybrid or Iron Bird item requires an attempt to reproduce the behavior on another simulator, or perhaps the Digital, in order to pinpoint the cause. When the cause of a discrepancy or anomaly is identified, an assessment is made to determine: (1) the operational impact when the problem is encountered if the program is left as is, (2) the procedures necessary to avoid or to work around the problem, (3) the coding change necessary to eliminate the problem, (4) the schedule impact of implementing and verifying the coding change. The assessment is documented as a PCR, PCN, or ACB which, if approved, is implemented as a fixed-coding change. Erasable coding is not used at this level for permanent changes. Disapproved PCRs, PCNs, and ACBs become program Notes. Sometimes it turns out that what was thought to be an anomaly, or discrepancy, was caused by a simulator bug, or a test deck error; in which case the problem is fixed and the test is rerun.

When all pending program changes are incorporated and tested at Level 4, and when no unresolved problems remain, the program is ready for release and is declared frozen. A technical review of the Level 4 testing is held (pre-FACI). If, in any areas the testing appears to need reinforcement, then new or additional Level 4 tests are defined. The Level 5 testing consists of rerunning all of the Level 4 test decks on the final version. If any new anomalies or discrepancies turn up and are serious enough to require a PCR, the Erasable Memory Program option is weighted heavily against a manufacturing schedule slip. The First Article Configuration Inspection (FACI) is a formal review of all Level 5 testing results, anomaly reports, change requests, program notes, and operational restrictions. The end action of the FACI is the granting of approval to release the rope assembly for manufacturing.

Flight Support Activity

The Flight Support Activity takes place after delivery of the Manufactured rope modules and centers around Level 6 testing as shown in Fig. 6. The KSTART tape is generated from the Preflight Erasable Load involving the Initial Data Load and any existing Erasable Memory Programs. Evaluation involves careful scrutiny of all parameters, by computer Program and by eyeball, to identify and assess changes from the previous KSTART tape. Additionally, the CSDL evaluation utilizes the Hybrid Simulator, the All-Digital Simulator, and the Systems Test Laboratory hardware installation. The testing is complemented by extensive mission-sequence testing on the Iron Bird Simulator at FRC, and involves pilot training, pilot procedures, and system performance. The test results are presented at the Flight Readiness Review (FRR), and any anomalies resolved, perhaps by modifying the operational envelope. FRR approval is required for flight go-ahead. Following a successful flight to test one DFCS capability, the Initial Data Load can be modified to test another capability, or to change the downlink coverage, and the procedure of Fig. 6 is repeated.

Alternatively, the flight test results can indicate a serious need for a DFCS capability that does not exist in the rope. In this case, a PCR is submitted to request that the capability be developed as an EMP. After assessment, if the PCR is approved, the development and test of the EMP is undertaken as was shown in the previous figure, Fig. 5. When completed, the verified EMP is incorporated into the KSTART tape for Level 6 testing.

Software Milestones

The development activity is tracked by milestones. Schedule milestones were not treated with the level of formality accorded their Apollo counterparts. Small meetings of one or two technical personnel with management personnel marked many F-8 DFBW events. Nevertheless, schedule milestones were vital to a timely development and verification process. The major milestones are indicated in Fig. 2.

The Preliminary Design Review (PDR) for F-8 consisted of several meetings, each covering a specific area of interest. These were preliminary in the sense that changes were expected as subcontractors and customer had the opportunity to review carefully each other's needs, plans, and suggestions.

The Critical Design Review (CDR) also consisted of several meetings, each covering a specific area in minute detail. The CDRs for the Control System Specification and the Interface Control Document are specific examples.

Level 1, 2, 3 Testing (Unit and Modular testing) allows tracking of units of software in the early stages of development when coding and verification are relatively independent of tight controls.

Configuration Control marks the transition to tightly controlled software configuration and testing procedures.

Level 4 Testing (Interface testing) allows tracking of interfaces between modules of software. Program changes require written approval and all anomalous simulation behavior requires documentation, analysis, and resolution.

Level 5 (Formal testing) allows tracking of software prototype.

First Article Configuration Inspection (FACI) is a formal review of all aspects of prototype software. The final action is the approval of the final assembly for manufacture.

Release-for-Manufacture—Following FACI approval, a weaving tape is generated from the final assembly to be used for core-rope manufacture.

Level 6 Testing (Mission Performance testing) is based on the KSTART tape for the particular flight. Evaluation consists of exercising the KSTART tape on the three CSDL Simulators and on the FRC Iron Bird System.

A Flight Readiness Review (FRR) is conducted prior to each flight. A statement from CSDL is required concerning its review on the Preflight Erasable Load and KSTART tape. The initial FRR had the longest agenda. The review assessed the flight readiness of the primary control system, the backup control system, the flight vehicle subsystems, to name a few. Known anomalies and their avoidance or work-around procedures were discussed. Erasable Memory Programs were explained, both functionally and operationally. The failure analysis studies were reviewed, as well as available documentation. Flight readiness reviews subsequent to the initial flight generally consider the current KSTART tape and any newly applicable areas.

SOFTWARE VERIFICATION

The software verification process is vital to the preparation of reliable high-quality software. A screening process is employed, whereby code is subjected to many tests representing many different situations. This approach to testing is one of diminishing returns: early tests show up most of the coding errors, but the later tests build confidence in the overall quality of the program assembly. Establishing the proper balance between insufficient and excessive verification testing is a critical task. Indeed, the verification process does not terminate with release-for-manufacture; it continues, in the hope of catching any remaining errors before they show up operationally with unexpected and perhaps dangerous consequences.

The verification process cannot be separated from the assembly control process, at least prior to release-for-manufacture. The ultimate quality and reliability of code depends heavily on the verification process. The attainment of the verification goals involves far more than the execution of high quality object code available near the end of the software development cycle. Facilities are required in the early stages of program development when the code available is of low quality and may not even be executable. In the early stages a benign and cooperative environment is required; it must provide a detailed

visibility into the execution of code. Simplified, but fast-operating environment algorithms are desirable. Extensive diagnostic capability is mandatory, involving both run-time and post-run software packages. As code quality is refined, the environment quality can be updated to include such factors as sensor errors and higher order effects. Ultimately the code should be exercised in a highly realistic environment including as much real hardware as possible.

Software Verification Facilities

Several distinct facilities were utilized during the DFCS verification process. The complementary nature of their unique capabilities is significant. Each has contributed to the DFCS quality, and by its absence would have affected the development adversely, mainly in terms of schedule, but perhaps even in terms of operational performance. CSDL has utilized the All-Digital Simulator, the Hybrid Simulator, and the System Test Laboratory facilities for the software development and verification activities. NASA/FRC has utilized the analog Stage 1 engineering simulation, the bench lashup Stage 2 hardware integration simulation, and the Stage 3 Iron Bird Simulator for the systems design, hardware integration, design verification, and pilot training/evaluation activities.

Each of these facilities has contributed to the overall success of F-8 DFBW, but certainly the significant contributions to system integration have come from the Stage 3 Iron Bird Simulator. It was on this facility that significant hardware integration problems were first encountered. The Stage 3 piloted simulations gave insight for design-change evaluation. Stage 3 permitted real-time demonstration of failure effects, and permitted engineering preliminary and final design. Stage 3 was used for much supportive software verification and essentially all of the system design verification. For the flight testing, where CSDL's verification role was supportive, the Stage 3 simulation was especially important as the primary design, verification, and training tool.

The All-Digital Simulator (ADS) at CSDL played the significant role in F-8 software design, development, and verification, primarily because of the powerful run-time diagnostic and post-run edit capability, as well as features such as repeatability and snapshot/rollback. Rigidly controlled simulator software provided a stable environment and ensured repeatability.

The Hybrid Simulator at CSDL was a very useful tool during preliminary verification, primarily because of its real-time interactive capabilities. Its role was somewhat diminished because CSDL did not have DFCS design responsibility, which is where the real-time interactive aspects of hybrid simulation can vastly improve the control-system designer's efficiency. However, on two separate occasions, one being the time-critical development period between Stage 2 and Stage 3 simulation, NASA/FRC came to CSDL and conducted basic and detailed design on our Hybrid facilities.

Piloted simulations early in the development phases can improve the overall quality of the end item, especially when schedules are tight. Pilot contributions cover a wide range of experience including such items as human

factors suggestions, functional change requests, performance and handling qualities evaluation, and safety considerations.

The complementary nature of all-digital, hybrid, and hardware integration facilities is important. The ADS provides diagnostic and edit capability plus detailed hard-copy for documentation. The Hybrid Simulator is unmatched in its real-time interactive capabilities for preliminary design, parameter-variation, and sensitivity studies. The hardware integration facility represents the ultimate interface verification tool short of flight test. Here, interfaces are actually mated, often for the first time. Failures can be studied and pilot-in-the-loop evaluations based on a maximum hardware complement can be performed. Each of the design, development, verification, and training tools can play a key non-overlapping role. It is the complementary nature of each facility which should be emphasized and utilized for greatest program efficiency and end-item quality.

A brief description of each of these facilities follows.

CSDL All-Digital Simulator—The Apollo Digital Simulator is a basic tool developed and employed primarily to support the design, development, and verification of Apollo Guidance Computer (AGC) programs. The simulator is entirely digital and consists of a number of programs implemented on a general purpose digital computer. It simulates the operation of the AGC in storage layout, and in detailed arithmetic and logical operation. Consistent with one's objectives, mathematical and logical models ranging from rudimentary to comprehensive may be selected to simulate the hardware and flight environment within which the AGC and its coding operate. For the F-8C, only the rigid body degrees of freedom are mechanized and there is no takeoff or landing capability. The BCS flight control system is not simulated, so controlled flight is possible only in the DFCS modes. The Pilot Action Simulator provides open-loop actions such as stick and rudder deflections, push button and trim switch activity, and DSKY operations. In addition, the simulator has numerous on-line diagnostic features, a snapshop/roll-back capability, and extensive post-run edit capability available. The edit package provides for flexible run-time data storage and for post-run data retrieval. The user has the choice of using standard edit programs or of writing his own. Extensive edit programs for plotting, computational verification, and formatting were developed for F-8 formal verification. Summary printing includes data on DFCS mode changes, timing, and computational delays. Plot variables include numerous DFCS and environmental quantities. Timing data indicating duty cycle and job activity is plotted. A downlink processor edit was prepared to verify proper downlink operation. The simulation system is illustrated schematically in Fig. 7.

The CSDL Hybrid Simulator—The Hybrid Simulator is a combination of selected flight hardware used in concert with analog and digital computers to provide real-time simulated flight. The flight hardware consists of an LGC computer, a DSKY, and the coupling data units. The LGC memory is replaced by a Core Rope Simulator (CRS), which provides a complete erasable memory as well as helpful features, such as the ability to monitor and change location contents, to stop at a location address, or to single-step the program. The IMU

is simulated with special-purpose electronics. Elements needing precision of storage, as the trajectory dynamics, the aerodynamics, and the rotational transformations, are simulated in an XDS 9300 digital computer. The high-frequency actuator dynamics, the BCS loops, and some discrete logic are simulated on the analog computer. The algorithms for BCS control and BCS downmode-trim initialization are simulated, but the cross-channel comparator and the hydrologic subsystems of the F-8C are not modelled. Also, provision is not made for a parking, landing, or takeoff capability. A minimal cockpit uses the Apollo three-axis rotational hand controller in place of stick/pedal controls. Cockpit instrumentation includes artificial horizon, altitude, airspeed, rate-of-climb, % thrust, g, angle of attack, and a mockup Mode And Power Panel for real-time man-in-the-loop simulations. Strip-chart recordings and initialization printout are the only hard-copy output. The Hybrid Simulator runs in real time to allow man-in-the-loop testing, on-line debugging, and flexibility in verification procedures. The LGC can function alone or with the Simulator providing an environment; in the former mode it is available independently of the availablity of the hybrid facility. Reproducibility is not in general possible, but this is an advantage in that a realistic randomness is introduced into the testing.

CSDL System Test Laboratory—The System Test Laboratory (STL) is an Apollo hardware integration facility. A real IMU interfaces with the LGC, CRS, and DSKY. Uplink and downlink are operational. Channel inbit discretes can be set or cleared manually and independently. The aircraft and BCS systems are not simulated. A trace capability is available via the Apollo CORONER and off-line processing; this is the only hard-copy output from this facility.

NASA/FRC Stage 1 Simulator—The Stage 1 Simulator was a preliminary design tool used to develop the flight control system specification equations. Simple analog models and sample-and-hold networks were utilized. Linear analysis based on continuous and sample-data control system design, using root locus and w-plane techniques, provided backup for the simulation effort.

NASA/FRC Stage 2 Simulator—The Stage 2 Simulator was a hardware integration and preliminary design evaluation facility. Breadboard lashup of major hardware components was first performed here. The LGC, the Program Analyzer Console (PAC, equivalent to the CRS), DSKY, IMU Gimbal Angle Simulator (GAS), and CDU package were involved. Aircraft and aero-surface servo actuator dynamics were modelled on a small analog computer. A rudimentary version of the DFCS and Operating System software participated.

NASA/FRC Stage 3 Simulator—The Stage 3 (or Iron Bird) Simulator is an F-8C airframe that includes all key hardware in the configuration of the flight article, including the pallet mounting of the LGC computer, IMU, and CDUs. The BCS electronics, power supplies, and hydraulics are flight-article type systems. The manufactured core-rope or PAC software can be used as the LGC memory. Simulated trajectory dynamics and aerodynamics permit closed-loop simulations using the GAS. Simple external visuals, sideslip angle and horizon line with sky/earth differentation, are provided on a TV screen mounted on the aircraft nose. Access to LGC and flight control system variables is by means of downlink with post-run editing or by DSKY display.

Software Verification Testing

It is difficult to separate software development and software verification since both go hand in hand throughout the development phase. To consider software verification it is necessary to consider software development. Generally speaking, there are two categories of software design changes that contribute to program construction.

(1) Developmental changes - these are creation of a new program or a new routine, or extensive changes within an existing program or routine.

(2) Incremental changes - these are modifications to existing code that cause small alterations and repercussions.

Clearly, a Developmental change has a major impact on the existing program and requires an extensive testing approach to assure that the new code works properly and does not interfere with other existing coding. It is equally clear that an Incremental change has a minor impact on the existing code and requires a localized testing approach. This is sort of by definition. However, it is not always clear into which of the two categories a given software change should be placed. Classification is a difficult problem and requires experience and thorough knowledge of the programs. For example, a one word change could require extensive testing if that word were, say, a sample period affecting event timing. On the other hand, the replacement of one Boolean relationship by another, involving perhaps 30 words, could be local in effect and require only local testing. Thus, the full arsenal of testing is brought to bear on Developmental software, while a subset is used for Incremental software.

Developmental Software Testing—In order to test out developmental changes, the six official levels of testing are normally performed. These are Unit testing (Levels 1 and 2), Modular testing (Level 3), Interface testing (Level 4), Formal testing (Level 5), and Mission Performance testing (Level 6). The majority of the F-8 DFBW programming effort falls into the developmental category, as exemplified by the flight control coding, input/output processing, ground test programs, and special routines. Design changes that occur late in the development cycle are often accorded the Developmental treatment. Erasable Memory Program design is also in this category, although there have been exceptions.

Incremental Software Testing—Incremental changes require adequate testing to assure that all paths in the program affected by the change are exercised. This may necessitate designing new tests for specific code changes. Incremental testing involves some combination of Unit testing, Modular testing, and Interface testing. Since all incremental changes become part of the program rope, they are automatically subjected to Level 5 and Level 6 testing.

There have been a number of incremental changes in F-8 DFBW. Initially, much of the software (about 60%) came from the Apollo Lunar Module Program. Many areas of the code required minor incremental changes to meet F-8C requirement: Late in the development cycle, especially as the release-for-manufacture date approached, changes even to flight control code can often be treated as incremental, especially if significant Level 4 interface testing has already been completed.

Some Erasable Memory Programs have been classified as Incremental. In one case, two lines of code were added to an existing EMP to create the one-pulse rudder pedal deadband. The other case was a preflight checkout program. These have received minimal Level 4/5 testing. Conversely, other EMPs involved significant design changes deeply imbedded in interface or systems code: parabolic shaping of stick inputs, or restart-triggering of BCS downmoding. These have received significant Level 4/5 testing, being developmental in nature.

Special Testing—There are a number of special tests deserving of mention that establish confidence in the flight software mainly by failing to find a fault rather than by exhaustively proving every possibility. This approach is in general true when the number of ways to exercise the code becomes unwieldy. The fact that interaction between the Executive, interrupt processors, and service routines falls into this category can be overlooked. A specific example is restart testing where a large number of artificially generated asynchronous time-triggered and location-triggered interrupts exercise the restart protection mechanism. Stress testing involves testing operational sequences under abnormal conditions. Potential anomaly testing attempts to duplicate the event sequences which led to questionable behavior on another hybrid facility. Hybrid testing occasionally encounters unexpected behavior that is usually a hardware problem, but can be a software problem. If a problem is found, digital testing gives conclusive evidence. Alternatively, if no problem is found, a measure of confidence is restored.

An 'eyeballing' effort was performed on the F-8 DBFW assembly just prior to release. Experienced Apollo programmers were assigned sections of the code to eyeball for errors, based on their accumulated experience. Several errors were uncovered, although off-nominal operational procedures would have been needed to encounter difficulties. The fact that errors were found gave weight to the effort as a worthwhile task. The absence of any serious errors, and the minimal number of errors encountered, added to the confidence level being built by the verification process.

Input and Output Discrete Failure Effects

A formal failure effects investigation was conducted late in the development cycle by CSDL and by other systems contractors. All interfaces were studied for fail-on and fail-off effects. Engineering analysis was the primary investigative tool, but simulated failures were utilized whenever pilot-in-loop problems were expected. To this end, a special version of the mainline program was created for the Iron Bird and was given the capability to fail any selected input/output discrete in the off-state or on-state. Failures were introduced during Iron Bird piloted simulations by a test engineer at the DSKY. The capability enabled pilot training in recognition and recovery procedures.

An important conclusion of the failure analysis is that such studies should be initiated early in the preliminary design phase so that failure effects can be recognized and avoided by careful design of hardware, software, and interfaces. Early recognition leads to design changes that often can be incorporated at no additional cost, whereas late recognition can be quite expensive.

Erasable Memory Programs

The concept of an Erasable Memory Program only has application in reference to a fixed memory computer when the capability to manufacture a new fixed memory is no longer available. Certainly, as long as the capability does exist, the redesign of a portion of the program or the inclusion of a new portion poses no particular problem even in a relatively mature program. In F-8 DFBW for example, the result of early Iron Bird simulations uncovered a hardware interface problem in that the anti-dropout filter in the CDU error counters interfered with restart recovery. Since the software was still under development, a straightforward redesign of the restart recovery routine was undertaken, including redevelopment and verification. On the other hand, when the ability to remanufacture the rope memory is gone, it is necessary to resort to an artifice, like erasable memory programming, if any change is to be incorporated into the program flow. If, however, one is dealing with a programmable memory computer, then post-release software changes are treated the same as pre-release software changes. The purpose of this section on EMPs then is to illustrate by example that sufficient cause for software changes can and will arise after program release, and to describe the F-8 DFBW experience.

Some of the late Stage 3 Iron Bird discoveries were not compatible with software development schedules, bound as they were by the anticipated shutdown of the core-rope manufacturing facilities. Erasable memory programming and major hardware changes were required instead. For example, piloted simulations in early 1972 indicated pilot-response problems with certain computer failures. The work-around concept was straightforward and a software change could have been made, except that the DFCS was no longer software; core-rope manufacture was under way. Fortunately, an Erasable Memory Program (EMP-001, Restart Downmoding to BCS) could do the job, so remanufacture was not necessary. However, the design and especially the verification tasks were much tougher for the EMP than they would have been for the fixed-memory equivalent, a characteristic of most erasable memory programming. Nevertheless, the flexibility provided by last-minute software changes represents a major selling point for digital flight control.

Design changes to minimize the effects of stick/pedal input quantization were not formalized until after the first flight. Hardware changes had been made earlier, prior to core-rope manufacture, but these proved to be inadequate. Again, an Erasable Memory Program (EMP-004, Parabolic Stick Shaping) provided an acceptable approach, but the fixed-memory equivalent would have been easier to design, develop, and verify. Also, the DFCS computational burden would have been lower with the equivalent fixed memory code, and operational aspects would have been simpler.

Problems do not always show up during the systems analysis and preliminary design phases, no matter how detailed the activity, but instead crop up during the hardware integration phase, or even worse, conceal their identity until the flight test phase. F-8C, during high-q flight for example, encountered a single-pulse null shift in the output from the pedal LVDT, which supplies the rudder pilot commands to the DFCS. The phenomenon apparently has something to do with

airframe distortion at high-q flight conditions. Neither the Stage 3 Iron Bird Simulator nor preliminary analysis models could indicate such a phenomenon. In this case, the hardware problem of rudder bias shift was eliminated by software, by inserting a one-pulse deadband (EMP-007, Single-pulse Pedal Deadband). There is a real motivation for a flight test phase, however brief, between the prototype and production software.

CONCLUDING REMARKS

The F-8 DFBW is an experimental digital fly-by-wire testbed flight control system, implemented with Apollo off-the-shelf hardware. Existing off-the-shelf software and software control techniques were dictated by hardware as well as manufacturing schedule limitations. Software design was bottom-up. Time-efficient code was important because of LGC speed. (Some of the techniques discussed would not be applicable for a modern, faster, all core computer.) Despite the LGC fixed memory, post-manufacturing design changes to the Specification were possible through Erasable Memory Programs. Proof of the benefits that accrue from good software control and from careful and thorough verification testing is evidenced by the F-8 DFBW flight test program results: In a year and a half, 42 flights, totaling 58 hours of flight time, were made successfully without any DFCS inflight software failures or performance surprises.

REFERENCES

The author has made generous use of References 1 and 2. Apollo hardware details are not included but can be found in Reference 2 and Reference 3. Supportive use was made of References 4, 5, and 6.

1. Engel, Albert G. Jr.,: F-8 Digital Fly-by-Wire Software Control/ Management, E-2749, Charles Stark Draper Laboratory,Cambridge, Mass., February 1973

2. Engel, Albert G. Jr.,: F-8 Digital Fly-by-Wire, Some Observations, E-2739, Charles Stark Draper Laboratory, Cambridge, Mass., January 1973

3. Miller, J. E., editor: Space Navigation, Guidance and Control. AGARDograph 105, August 1966

4. Hamilton, M. H.: Management of APOLLO Programming. Mission Program Development Note 18, Charles Stark Draper Laboratory, Cambridge, Mass., May 1971

5. Johnson, M. S. and Griller, D. R.: MIT's Role in Project Apollo, Volume V, The Software Effort, Charles Stark Draper Laboratory, Cambridge, Mass., July 1971

6. David, S. S., editor: Users Guide to the Apollo Digital Simulator, Charles Stark Draper Laboratory, Cambridge, Mass., April 1972

TABLE 1

APOLLO HARDWARE USED IN F-8 DFBW

LGC	-	LM Guidance Computer (approximately 2k of erasable and 36k of programmable fixed core-rope memory; programmable hardware-interrupt and software-executive systems; hardware restart logic, etc.).
DSKY	-	(LM) Display and Keyboard (three 5-digit-plus-sign display windows; miscellaneous warning lights; keyboard including 0 through 9, +, -, PRO (proceed), ENTR, CLR (clear), VERB, NOUN, etc; the DSKY is the computer/astronaut or computer/ground crew interface).
IMU	-	Inertial Measurement Unit (a three-gimballed gyroscopically-stabilized platform for the PIPA accelerometers; gimbal angle resolver and PIPA signals ultimately interface with the LGC; several platform alignment techniques are under LGC software control).
CDU	-	Coupling Data Unit (for analog-to-digital conversion of IMU gimbal angle indications; for digital-to-analog conversion of LGC computer outputs; for control of IMU moding; includes failure detection; used to derive body axis angular rates).
PIPA	-	Pulsed Integrating Pendulous Accelerometer (three mutually-perpendicular contact-acceleration-sensing and incremental-velocity-indicating devices located on the IMU stable member, with a direct LGC interface; used to derive body axis normal and lateral acceleration).
PSA	-	Power and Servo Assembly (power supplies, amplifiers, etc., for inertial subsystem).
PTA	-	Pulse Torque Assembly (input/output processing for inertial subsystem).

TABLE 2

HARDWARE UNIQUE TO F-8 DFBW

MAPP	-	Mode and Power Panel (computer and IMU power control, autopilot gain and mode select/indicators, warning indicators, etc.
IFB	-	Interface Box (junction box containing an Apollo DAC stick/pedal comparators, special amplifiers, etc.).
BCS	-	Backup Control System (triply-redundant stick/pedal to aerosurface open-loop control, with trim, hydrologic comparator; cross-channel comparator; etc.).
DLC/IFR	-	Downlink Converter/Inflight Recorder (100 word-pair list every 2 seconds on a 20ms interrupt; recording on FM tape for post-flight processing/review).
GSE	-	Ground Support Equipment (the Apollo Program Analyzer Console (PAC) for simulating LGC hard-wire rope memory; the Uplink Converter (ULC) for preflight erasable loading and for DSKY-type program control via tape; the Ground Test Cart containing downlink converter/ground recorder, miscellaneous switches and indicators; etc.).
SPCC	-	Servo Pressure Control Console (PRI select/indicators for each axis; servo pressure switches and indicators for each BCS servo-valve and for PCS servo-valve pairs; each switch has three positions: OFF which disables that valve, AUTO which enables that valve, and MAN which overrides any automatic moding and locks that valve into the active state).
CCS	-	Coolant Control System (coolant for IMU, computer, etc.).

TABLE 3

F-8 DFBW FIXED-MEMORY ALLOCATION

F-8 DFBW Flight Control System (total)		5586
Body Rate/Acceleration Feedback	320	
Generalized Feedback Filters	1930	
Pilot Stick/Pedal Processing	168	
Control Loop Equations	1178	
Channel Monitor Routine	523	
Gain/Mode Change Routine	985	
Initialization/Restarts/Miscellaneous	482	
Ground Test Programs/Extended Verbs		768
Self Test/Check		1436
Fresh Start/Restart/V37/etc.		853
Display Interfaces/Pinball/etc.		3578
Interpreter/Executive/Waitlist/Downlink/Uplink/etc.		3830
IMU Alignment, Compensation, and Tests/T4RUPT		3263
TOTAL F-8 DFBW FIXED-MEMORY USED		19314
TOTAL LGC FIXED-MEMORY (36 FBANKS AT 1024)		36864

TABLE 4

F-8 DFBW ERASABLE-MEMORY ALLOCATION

Preflight Erasable Load (total)		389
F-8 DFBW Flight Control System	169	
IMU Compensation/Alignment	33	
Erasable Downlink List	100	
Erasable Memory Programming (EMP-001,4,7)	87	
F-8 DFBW Flight Control System Working Registers		321
Extended Verbs/Ground Test Prog/Miscellaneous		50
Self Test/Check		263
IMU Alignment/Perf Test/Ops Test		17
Uplink/Downlink		32
Display Interfaces/Pinball/etc.		56
Executive/Waitlist/Service/Centrals/etc.		468
TOTAL F-8 DFBW ERASABLE-MEMORY USED		1596
TOTAL LGC ERASABLE-MEMORY (8 EBANKS AT 256)		2048

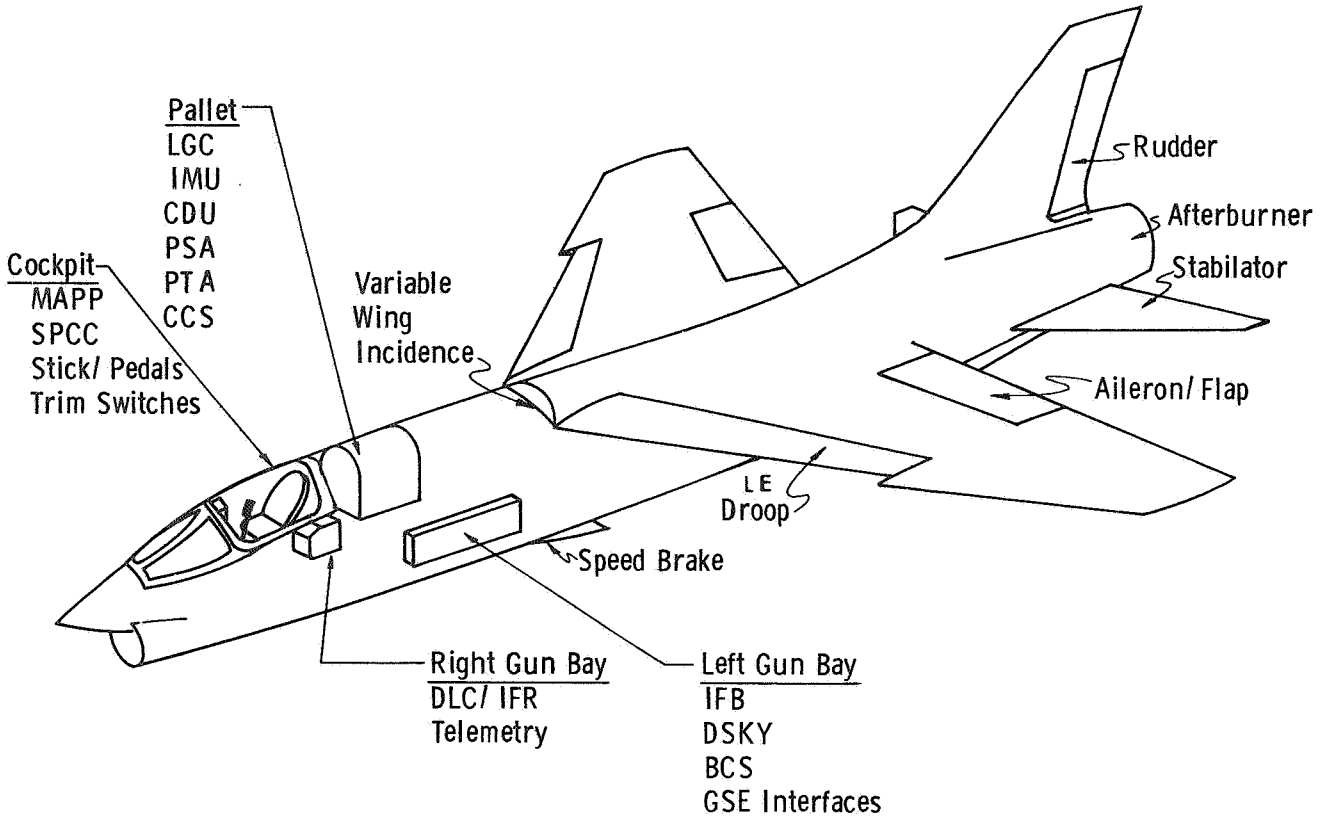

Fig. 1. F-8C DFBW Aircraft and Hardware

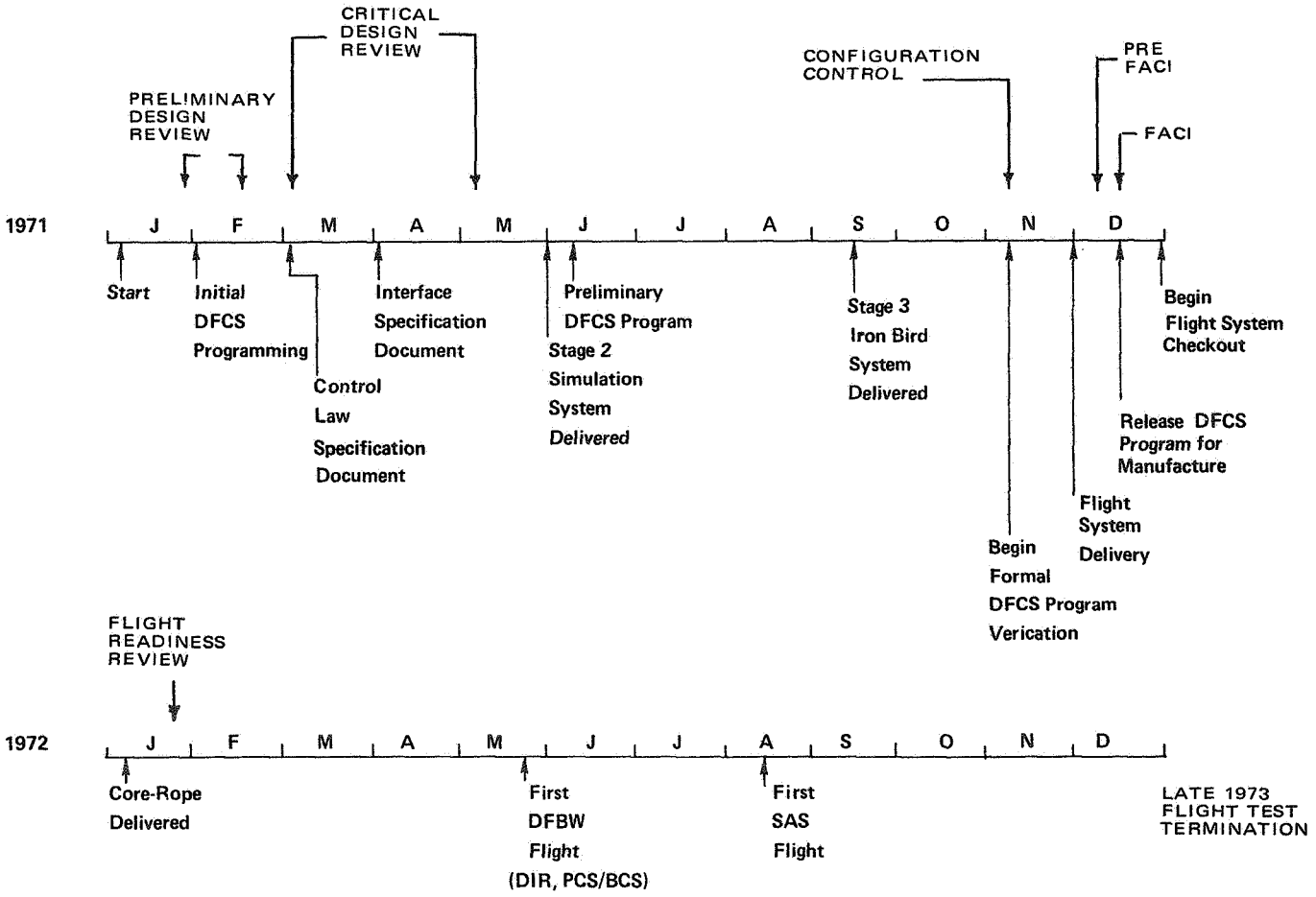

Fig. 2. F-8 DFBW Chronology

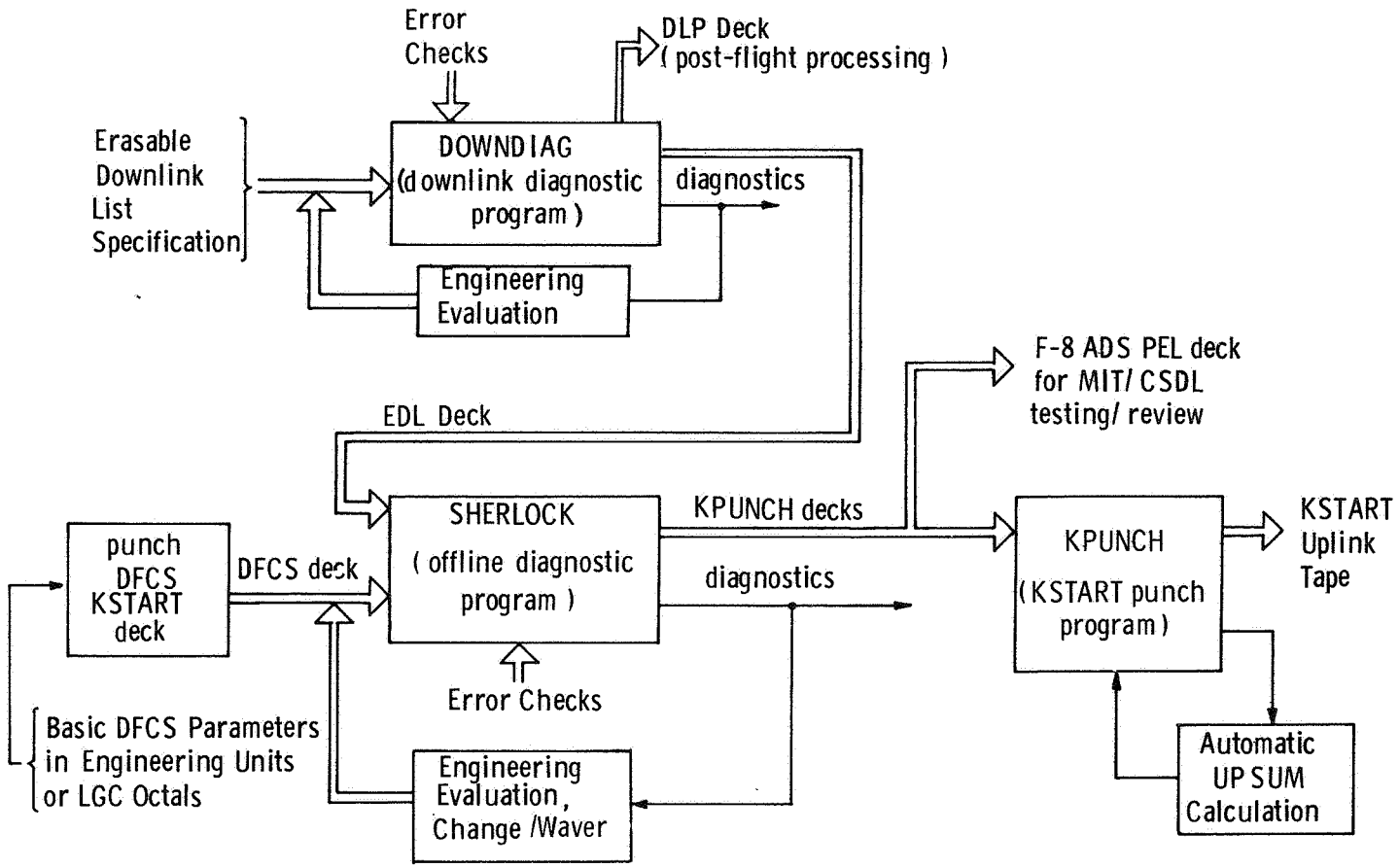

Fig. 3. KSTART Preparation

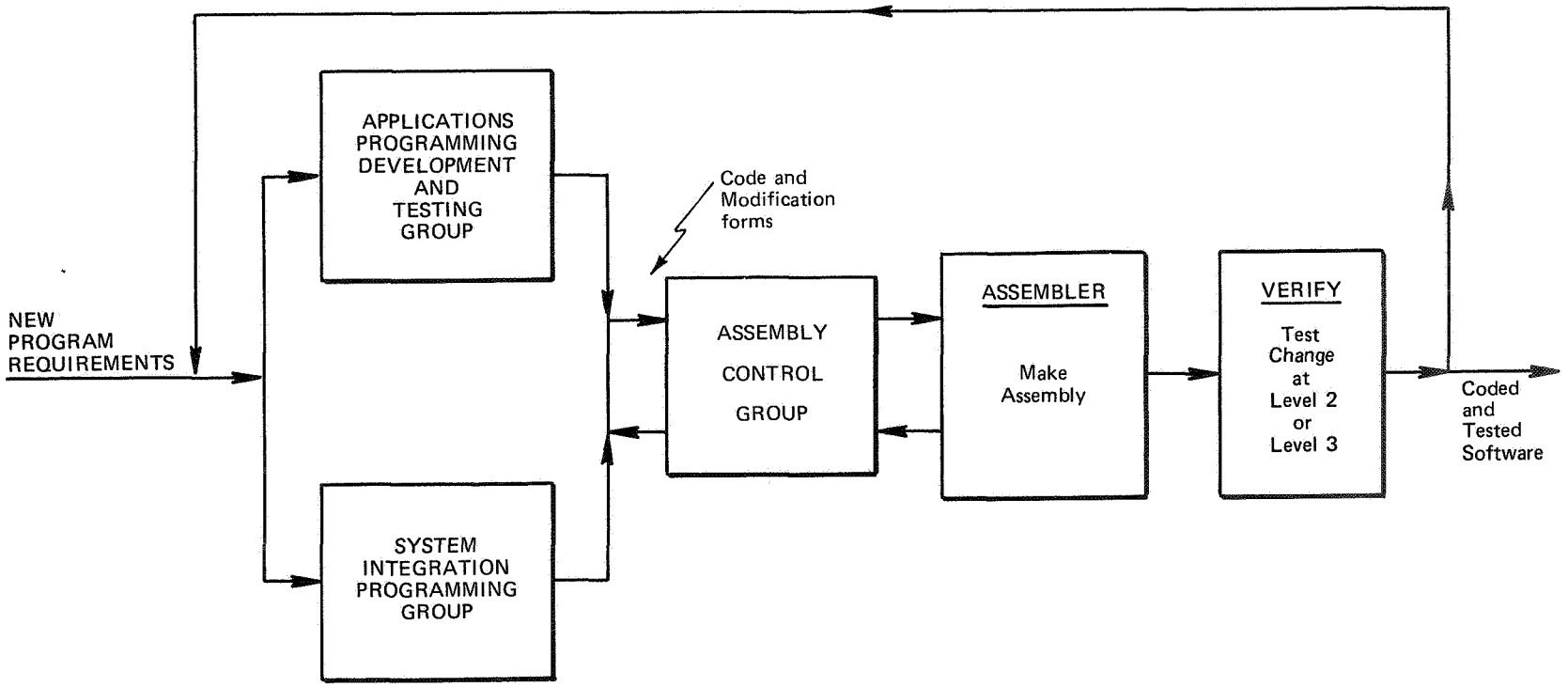

Fig. 4. Program & Assembly Control Activity

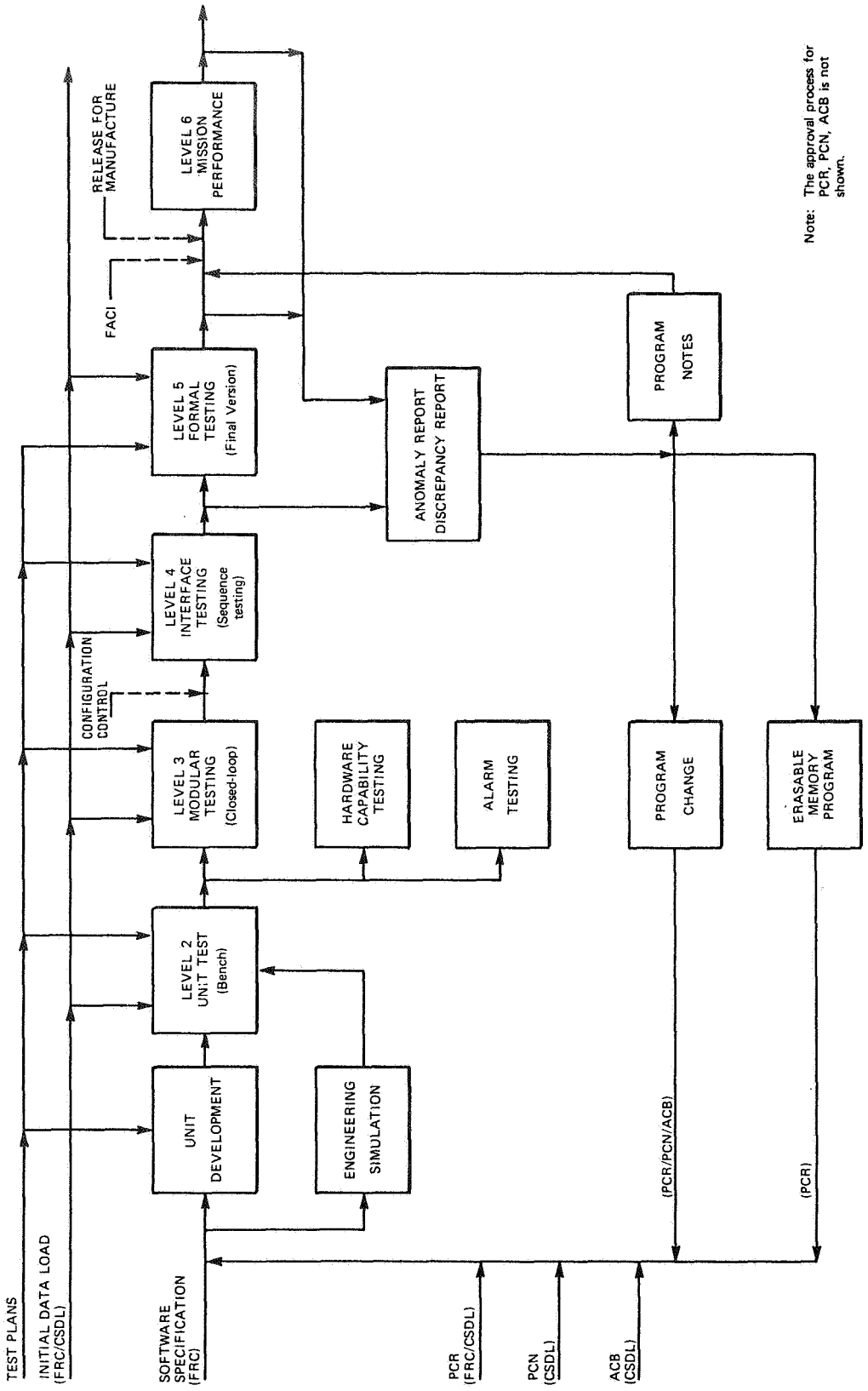

Fig. 5. F-8 DFBW Software Development Activity

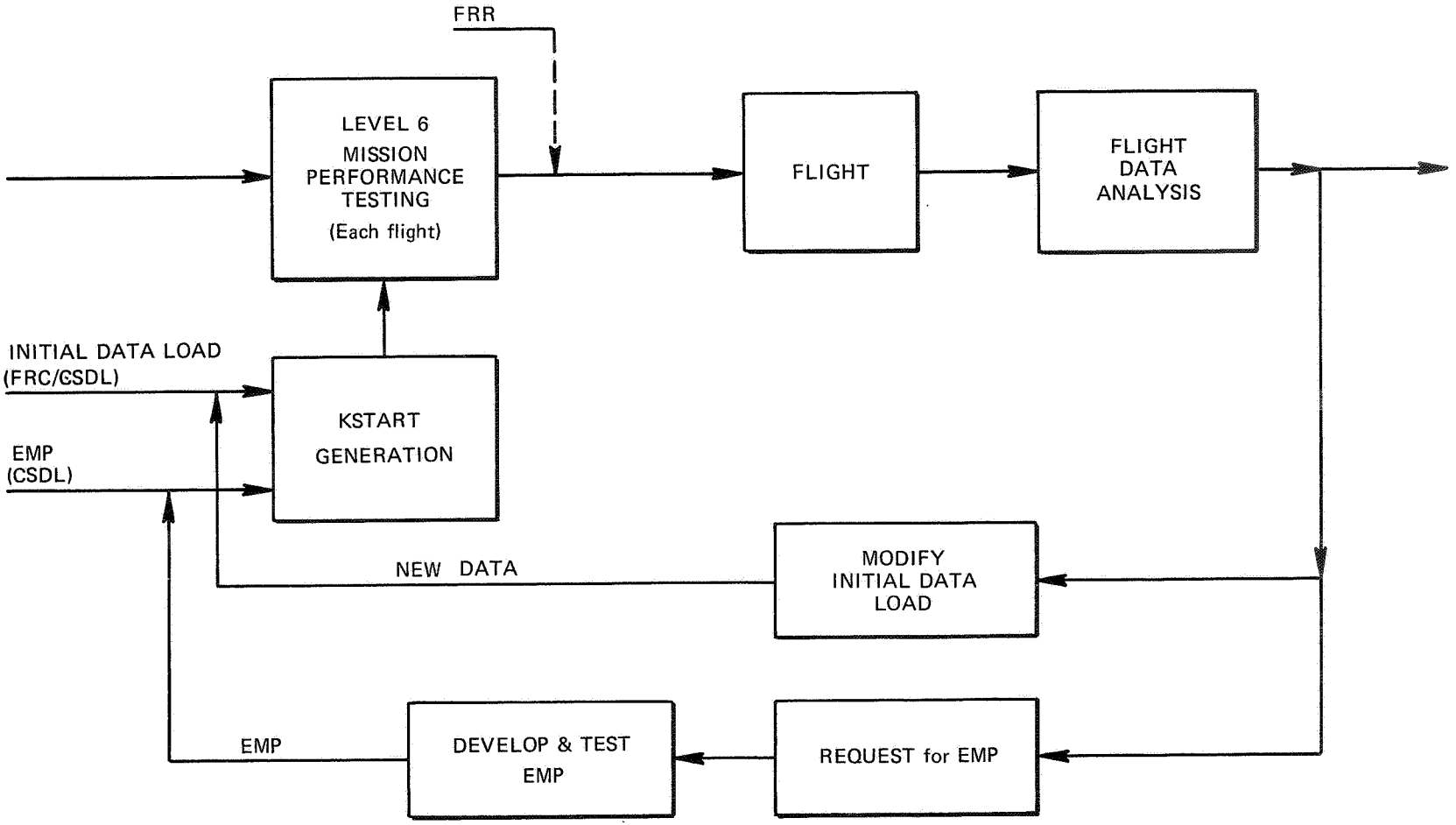

Fig. 6. Flight Support Activity

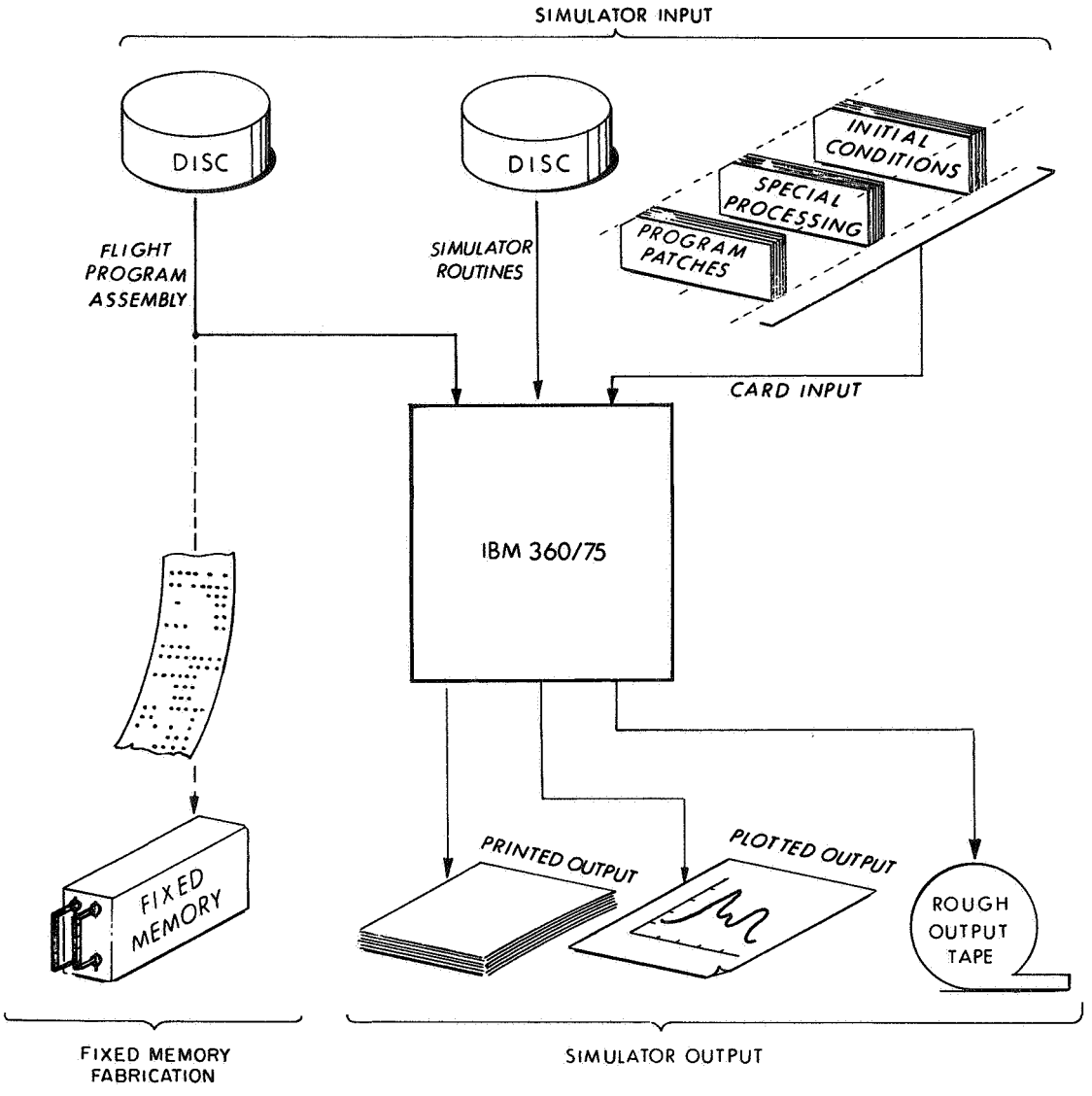

Fig. 7. Simulator System Schematic